Theoretical Computer Science 16 (1981) 223–229
North-Holland Publishing Company

NOTE

TESTING FOR THE CHURCH–ROSSER PROPERTY*

Ronald V. BOOK and Colm P. O'DUNLAING
Department of Mathematics, University of California, Santa Barbara, CA 93106 U.S.A.

Communicated by M. Nivat
Received February 1981

Abstract. It is shown that there is a polynomial time algorithm to determine whether a finite Thue system is Church–Rosser.

1. Introduction

Replacement systems arise in the study of formula manipulation systems such as theorem provers, program optimizers, and algebraic simplifiers, where they take the form of term rewriting systems, tree manipulating systems, graph grammars, etc. Usually the problem is to define some efficient operational semantics for an equational theory. Such systems are also useful in the study of abstract data types since this same type of problem arises there. Often one attempts to show that the system is 'confluent' or 'Church–Rosser' so that there is a way of describing canonical representatives or unique normal forms. See [4, 6, 7, 9, 12].

When dealing with strings, the appropriate notion of replacement system appears to be that of Thue system, especially in the case when the strings in questions are 'unstructured', i.e., taken from an arbitrary free monoid on a finite alphabet. The question of whether an infinite Thue system is confluent or Church–Rosser is undecidable even when such a system is finitely specified [3]. On the other hand, the question of whether a finite Thue system is confluent is decidable, a result due to Nivat [11]. The result established by Nivat leads immediately to an exhaustive search algorithm that determines whether the given system is confluent. In this paper Nivat's result is extended to show that it is decidable whether a finite Thue system is Church–Rosser. Further, it is shown that exhaustive search is not necessary and that these questions are tractable, i.e., there is a polynomial time algorithm to decide this question (see [5]). The algorithm presented here depends on two facts:

* This research was supported in part by the National Science Foundation under Grant No. MCS80-11979.

0304-3975/81/0000–0000/$02.50

(i) in a Church–Rosser Thue system two different irreducible strings cannot be congruent; and

(ii) for any finite Thue system there is a linear time algorithm to compute from a given string x an irreducible string y such that y is congruent to x [2].

2. Thue systems

If Σ is a finite alphabet, then Σ^* is the set of all strings over Σ with the empty string e. If $w \in \Sigma^*$, then $|w|$ denotes the length of w.

A *Thue system* S on a finite alphabet Σ is a set of ordered pairs of strings from Σ^* called *rules*. The *Thue congruence* defined by S is the reflexive transitive closure $\leftrightarrow^*_{(S)}$ of the relation $\leftrightarrow_{(S)}$ defined as follows: If $(u, v) \in S$, then for all x, y, $xuy \leftrightarrow_{(S)} xvy$ and $xvy \leftrightarrow_{(S)} xuy$. (The subscript S will be omitted whenever possible.)

Throughout this paper it is assumed that if S is a Thue system and $(u, v) \in S$, then $|u| \geqslant |v|$. Since the relation $\leftrightarrow_{(S)}$ is symmetric, no generality is lost by making this assumption.

If S and T are Thue systems with the property that for all x, y, $x \leftrightarrow^*_{(S)} y$ if and only if $x \leftrightarrow^*_{(T)} y$, then S and T are *equivalent*.

If S is a Thue system, write $x \to y$ if $x \leftrightarrow y$ and $|x| > |y|$, and write $x \vdash y$ if $x \leftrightarrow y$ and $|x| = |y|$; let $\to^*(\vdash^*)$ be the reflexive transitive closure of $\to$ (resp., $\vdash$). The relation $\to^*$ is referred to as a *reduction*.

Let S be a Thue system.

(a) S is *confluent* if for every choice of w, x, y, $w \to^* x$ and $w \to^* y$ imply that for some z, $x \to^* z$ and $\to^* z$.

(b) S is *Church–Rosser* if for every choice of x, y, $x \leftrightarrow^* y$ implies that for some z, $x \to^* z$ and $y \to^* z$.

When studying replacement systems, the reduction relation $\to$ may be given and then the relation $\leftrightarrow$ may be defined by $\leftrightarrow = \to \cup \to^{-1}$. This is not the case for Thue systems since there is the possibility of length-preserving rules so that $\leftrightarrow = \to \cup \to^{-1} \cup \vdash$. While a Thue system that is Church–Rosser is confluent [4, 6, 9], the presence of length-preserving rules may prevent a confluent Thue system from being Church–Rosser. However, if S is a Thue system such that $x \vdash y$ implies that for some z, $x \to^* z$ and $y \to^* z$, and such that S is confluent, then S is Church–Rosser and S is equivalent to $S - \{(u, v) \mid |u| = |v|\}$. Since $x \vdash y$ if and only if $x = wuz$ and $y = wvz$ where $|u| = |v|$ and $(u, v) \in S$, we have the following fact.

Lemma 1. *Let S be a Thue system. S is Church–Rosser if and only if both of the following conditions hold:*

(a) *For every $(u, v) \in S$ such that $|u| = |v|$, there exists z with $u \to^* z$ and $v \to^* z$;*

(b) *The subsystem $S - \{(u, v) \mid |u| = |v|\}$ is confluent.*

Clearly, a Thue system with no length-preserving rules is confluent if and only if it is Church–Rosser.

Let S be a Thue system. A string x is *irreducible* (mod S) if there is no y such that $x \to y$; otherwise, x is *reducible* (mod S).

Irreducible strings are 'normal forms' for congruence classes. When a Thue system is Church–Rosser, normal forms are unique.

Lemma 2 ([4, 6, 7, 9]). *Let S be a Church–Rosser Thue system.*

(a) *If x and y are irreducible and $x \leftrightarrow^* y$, then $x = y$.*

(b) *For every x there is a unique irreducible y such that $x \leftrightarrow^* y$.*

A possible difficulty in carrying out transformations in a Thue system S is that there may exist rules (u, v_1) and $(u, v_2) \in S$ with $v_1 \neq v_2$. If S is Church–Rosser, it is sufficient to make arbitrary choices.

Lemma 3. *Let S be a Thue system. Suppose that $T \subseteq S$ is any subsystem of S with the properties*

(a) *T has no length-preserving rules, and*

(b) *for any string u such that for some v, $|u| > |v|$ and $(u, v) \in S$, there is a unique v' such that $|u| > |v'|$ and $(u, v') \in T$.*

If S is Church–Rosser, then T is Church–Rosser and T is equivalent to S. Further, a string is irreducible (mod T) *if and only if it is irreducible* (mod S).

Proof. The fact that a string is irreducible (mod T) if and only if it is irreducible (mod S) follows immediately from property (*b*). (This fact does not depend on S being Church–Rosser.)

Suppose that S is Church–Rosser. We claim that for every x, z, if $x \to^* z$ (mod S) and z is irreducible, then $x \to^* z$ (mod T). To prove this fact, notice that if $x \to^*_{(S)} z$ and z is irreducible (mod S), and $x \to^*_{(T)} y$ and y is irreducible (mod T), then y is irreducible (mod S) and so $y = z$ since S is Church–Rosser. Thus, $x \to^*_{(T)} z$.

If $x \leftrightarrow^* y$ (mod S), then since S is Church–Rosser, there exists an irreducible z such that $x \to^* z$ (mod S) and $y \to^* z$ (mod S). By the above, $x \to^* z$ (mod T) and $y \to^* z$ (mod T) so $x \leftrightarrow^* y$ (mod T). Since T is a subsystem of S, this implies that T is equivalent to S.

If $x \leftrightarrow^* y$ (mod T), then $x \leftrightarrow^* y$ (mod S). Since S is Church–Rosser, there exists an irreducible z such that $x \to^* z$ (mod S) and $y \to^* z$ (mod S). By the above, $x \to^* z$ (mod T) and $y \to^* z$ (mod T). Hence, T is Church–Rosser.

3. The algorithm

Nivat [11] has shown that a Thue system S is confluent if and only if the following condition holds: For every pair of (not necessarily distinct) rules (u_1, v_1), (u_2, v_2) in S with $|u_1| > |v_1|$ and $|u_2| > |v_2|$,

(a) if there exist x, y such that $u_1 x = y u_2$ and $|x| < |u_2|$, then there exists z such that $v_1 x \to^* z$ and $y v_2 \to^* z$; and

(b) if there exist x, y such that $u_1 = x u_2 y$, then there exists z such that $v_1 \to^* z$ and $x v_2 y \to^* z$.

The result of Nivat is related to the strategy of Knuth and Bendix [8]. In the case of Thue systems, one deals only with strings of 'constants', not variables, and so certain tasks become easy. See [6, 7] for further discussion.

If S is a finite Thue system, then one can test S to see if Nivat's condition holds (since the number of rules in S is finite), and so the question of whether a finite Thue system is confluent is decidable [11].

A simple proof of Nivat's result follows from the fact that a Thue system S is confluent if and only if for all w, x, y, $w \to x$ and $w \to y$ implies that for some z, $x \to^* z$ and $y \to^* z$ (in fact, z can be assumed to be irreducible). This fact was established by Newman [9] for replacement systems arising in combinatorial topology and later was established by Huet [6] for abstract replacement systems. Huet's proof is quite simple and uses the principle of Noetherian induction.

Here we describe an algorithm to perform Nivat's test for confluence, adding one additional condition so as to have an algorithm to test for the Church–Rosser property. Input to the algorithm will be a finite set S of ordered pairs (u, v) of strings with $|u| \geq |v|$. Let $|S| = \Sigma\{|uv| \mid (u, v) \in S\}$. The algorithm will operate in stages and each stage will run in time $O(|S|^6)$.

Stage 1: Initialization

Read the input S, storing the left-hand side u of each rule in a table which can be used to access in a linked list all the corresponding right-hand sides v such that $(u, v) \in S$. Store the right-hand sides in lexicographic order. Determine a subsystem $T \subseteq S$ by considering each left-hand side u and choosing the first right-hand side v such that $|u| > |v|$ and $(u, v) \in S$. If such a v exists, put $(u, v) \in T$; otherwise, there is no v such that $(u, v) \in T$.

Once T is determined, construct a 'trie' by which the reversal u^R of strings u occurring in the left-hand side of some rule in T (not S) may be quickly recognized. This is essentially a finite-state automaton accepting a finite set of strings. Proceed to Stage 2.

The links from each node of the trie can be stored as a list so that during the REDUCE routine (see below) in $|\Sigma|$ steps one can determine whether a symbol popped matches a link. Restoring symbols requires a constant number of steps if each node has a back pointer to its parent, plus the symbol labeling the link involved. For fixed Σ the trie can be used to search for a longest match in linear time and occupies linear space, ignoring storage size for pointers and the time required to obtain the next element on the list. If Σ is considered as a parameter, then its size must be considered as a factor of the running time.

Assuming unit cost storage access and dynamic allocation, one can see that it only requires linear time and space to add a rule (u, v) to the trie, so that the trie can be constructed in linear time and space in Stage 1.

Subroutine: *REDUCE*

It will be useful to separate one specific task as a subroutine. The purpose of this subroutine is to compute from input x an irreducible string z such that $x \to^* z$ (mod T). Given a fixed system such that $(u_1, v_1) \neq (u_2, v_2)$ implies $u_1 \neq u_2$, one may construct a two-stack automaton which on input x generates an irreducible z such that $x \to^* z$ and which runs in time $k|x|$ where k depends on the size of the rules in the system [2]. The automaton can be simulated by using the trie from Stage 1 which supplies all the necessary information about T, and occupies space linear in the size of T. Hence we consider T as a parameter and so the algorithm must be considered to run in time that is quadratic rather than linear.

We refer to this algorithm as REDUCE and write $z :=$ REDUCE(x). REDUCE uses two stacks, stack 1 and stack 2. Initially, stack 1 is empty and stack 2 contains the input x with its leftmost symbol on top. When stack 2 is empty, stack 1 will contain the desired result. At any point between the initial and final steps, stacks 1 and 2 contain an intermediate string y such that $x \to^* y$ (mod T) with a prefix of y stored on stack 1 and the corresponding suffix of y stored on stack 2.

REDUCE has two phases, READ and SEARCH. Initially, a READ is attempted.

READ: Attempt to pop a symbol from stack 2 and push it onto stack 1. If stack 2 is empty, halt; otherwise go to SEARCH.

SEARCH: Using the trie for T, pop from stack 1 the longest possible word u (if any) which occurs as the left-hand side of a rule $(u, v) \in T$. If no such word is detected, restore stack 1 to its previous condition and go to READ. If such a string u is detected, push the unique string v to stack 2, where $(u, v) \in T$, and go to READ.

Notice that the leftmost symbol of the current intermediate string is on the bottom of stack 1 while the rightmost is on the bottom of stack 2. Thus in SEARCH the string u is read from right to left and the string v is pushed rightmost symbol first, and so the trie must store the reversal u^R of u.

To see that REDUCE does indeed compute an irreducible descendant of the input string, see [2]. To consider the cost of REDUCE, notice that both stacks contain at most $|x|$ symbols between them at any time so that only $O(|x|)$ space is needed (ignoring the size of the trie). Recall that $(u, v) \in T$ implies $|u| > |v|$. If u is popped from stack 1 and v is pushed onto stack 2 during an application of SEARCH, then $|u|$ applications of READ were needed to write u on stack 1 and so the entire process of "write u on stack 1, pop u from stack 1, push v onto stack 2" has the effect of reducing the length of stack 2 by $|u| - |v| > 0$. Thus we see that READ is invoked $O(|x||S|)$ times. Between READs, SEARCH can do no worse than to empty and refill stack 1 and push a shorter string onto stack 2, and so at most $2|x|$ steps are involved. Use of the trie can be regarded as taking a constant number of steps between successive extractions of a symbol from stack 1 if Σ is fixed, otherwise a number of steps bounded by $|S|$. Hence the overall time is $O(|x|^2|S|)$ when Σ is fixed.

Stage 2: *Length-preserving rules*

Process in turn each string u occurring as the left-hand side of at least one rule of S. Having selected u, consider each v such that $(u, v) \in S$ and $|u| = |v|$. Compute

$z_1 :=$ REDUCE(u) and $z_2 :=$REDUCE(v). Determine whether z_1 equals z_2. If for any such u and v, z_1 is not equal to z_2, then halt and report that S is not Church–Rosser. Otherwise, go to Stage 3.

From Lemmas 1–3, notice that if $(u, v) \in S$, $|u| = |v|$, and S is Church–Rosser, then there is a unique irreducible z such that $u \to^* z \pmod T$ and $v \to^* z \pmod T$. If S is Church–Rosser, then REDUCE(u) is the unique irreducible descendant of u and REDUCE(v) is the unique irreducible descendant of v. Thus Stage 2 correctly determines whether S has this property.

There are at most $|S|$ rules to be considered and each call of REDUCE takes at most $O(|x|^2|S|)$ steps for $x = u$ or $x = v$, so this portion of Stage 2 runs in time $O(|S|^4)$. Sequential search can be used to locate the various u and v and this can be accomplished in time $O(|S|)$.

Stage 3: *Testing for confluence*

For every pair (u_1, u_2) of (not necessarily distinct) strings that both appear as left-hand sides of rules in S, perform the following two tasks.

(i) Form all possible strings x, y, not both empty, such that $u_1 = xu_2y$. For each such x, y, compute REDUCE(v_1) and $z_2 :=$ REDUCE(xv_2y) where $(u_1, v_1) \in T$ and $(u_2, v_2) \in T$. Determine whether z_1 is equal to z_2. For any such x, y, if $z_1 \neq z_2$, then halt and report that S is not Church–Rosser.

(ii) Form all possible strings x, y such that $u_1x = yu_2$ and $0 < |x| < |u_2|$. For each such x, y compute $z_1 :=$ REDUCE(v_1x) and $z_2 :=$ REDUCE(yv_2), where $(u_1, v_1) \in T$ and $(u_2, v_2) \in T$. Determine whether z_1 is equal to z_2. For any such x, y, if $z_1 \neq z_2$, then halt and report that S is not Church–Rosser. If Stage 3 is successfully completed, then halt and report that S is Church–Rosser.

This stage terminates since in task (i) there can be at most $|u_1|$ such x, y pairs and in task (ii) there can be at most $|u_1| + |u_2|$ such x, y pairs. From Lemmas 2 and 3 it is clear that Stage 3 correctly checks that Nivat's conditions for confluence are obeyed. From Lemmas 1 and 3 it is clear that Stages 2 and 3 correctly check for the Church–Rosser property.

The number of pairs (u_1, u_2) is bounded by $O(|S|^2)$. Task (i) can call REDUCE at most $|u_1|$ times and task (ii) can call REDUCE at most $|u_1| + |u_2|$ times. Thus the total time in which Stage 3 uses REDUCE is at worst $O(|S|^6)$. Selecting all pairs (u_1, u_2) takes at most time $O(|S|^2)$ by sequential search and with a naive enumeration algorithm with a cost of $|u_1u_2|^2$ one can determine all x, y such that $u_1 = xu_2y$, and all x, y such that $u_1x = yu_2$ and $|x| < |u_2|$. Thus that part of Stage 3 requires at most time $O(|S|^3)$ and clearly linear space is adequate.

We state our result as follows:

Theorem. *There is a polynomial time algorithm to determine whether a finite Thue system is Church–Rosser.*

Thus, the problem of deciding whether a finite Thue system is Church–Rosser is tractable (see [5]). We make no claims about the 'best bound' for this question.

References

[1] J. Berstel, Congruences plus que parfaites et langages algébriques, *Seminarie d'Informatique Théorique*, Institut de Programmation (1976–77) 123–147.

[2] R. Book, Confluent and other types of Thue systems, *J. ACM.* to appear.

[3] R. Book, M. Jantzen and C. Wrathall, Mondadic Thue systems, *Theoret. Comput. Sci.*, to appear.

[4] Y. Cochet and M. Nivat, Une généralization des ensembles de Dyck, *Israel J. Math.* **9** (1971) 389–395.

[5] M. Garey and D. Johnson, *Computers and Intractability* (Freeman, San Francisco, 1979).

[6] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems, *18th IEEE Symp. Foundations Comp. Sci.* (1977) 30–45. Also, *J. ACM.* **27** (1980) 797–821.

[7] G. Huet and D. Oppen, Equations and rewrite rules: a survey, in: R. Book, Ed., *Formal Language Theory: Perspectives and Open Problems* (Academic Press, New York, 1980) 349–405.

[8] D. Knuth and P. Bendix, Simple word problems in universal algebras, in: J. Leech, Ed., *Computational Problems in Abstract Algebra* (Pergamon Press, Elmsford, NY, 1970) 263–297.

[9] M.H.A. Newman, On theories with a combinatorial definition of 'equivalence', *Annals Math.* **43** (1942) 223–243.

[10] M. Nivat, On some families of languages related to the Dyck languages, *2nd ACM Symp. Theory Computing* (1970) 221–225.

[11] M. Nivat (with M. Benois), Congruences parfaites et quasi-parfaites, Seminaire Dubreil, 25^{e} Année (1971–72) 7–01–09.

[12] B. Rosen, Tree manipulating systems and Church–Rosser systems, *J. ACM* **20** (1973) 160–187.